PATENT COOPERATION TREATY

PCT

REC'D 0 1 JUL 2005
WIPO PCT

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY
(Chapter II of the Patent Cooperation Treaty)

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference 156066.1/Le/sc	**FOR FURTHER ACTION**	See Form PCT/IPEA/416
International application No. PCT/EP2004/050776	International filing date *(day/month/year)* 12.05.2004	Priority date *(day/month/year)* 05.06.2003
International Patent Classification (IPC) or national classification and IPC G06F9/40		
Applicant SWISS REINSURANCE COMPANY et al.		

1. This report is the international preliminary examination report, established by this International Preliminary Examining Authority under Article 35 and transmitted to the applicant according to Article 36.
2. This REPORT consists of a total of 9 sheets, including this cover sheet.
3. This report is also accompanied by ANNEXES, comprising:
   a. ☒ *sent to the applicant and to the International Bureau)* a total of 5 sheets, as follows:
      ☒ sheets of the description, claims and/or drawings which have been amended and are the basis of this report and/or sheets containing rectifications authorized by this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions).
      ☐ sheets which supersede earlier sheets, but which this Authority considers contain an amendment that goes beyond the disclosure in the international application as filed, as indicated in item 4 of Box No. I and the Supplemental Box.
   b. ☐ *(sent to the International Bureau only)* a total of (indicate type and number of electronic carrier(s)) , containing a sequence listing and/or tables related thereto, in computer readable form only, as indicated in the Supplemental Box Relating to Sequence Listing (see Section 802 of the Administrative Instructions).

4. This report contains indications relating to the following items:

☒	Box No. I	Basis of the opinion
☐	Box No. II	Priority
☐	Box No. III	Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
☐	Box No. IV	Lack of unity of invention
☒	Box No. V	Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
☐	Box No. VI	Certain documents cited
☐	Box No. VII	Certain defects in the international application
☐	Box No. VIII	Certain observations on the international application

Date of submission of the demand	Date of completion of this report
11.12.2004	30.06.2005
Name and mailing address of the international preliminary examining authority: European Patent Office D-80298 Munich Tel. +49 89 2399 - 0 Tx: 523656 epmu d Fax: +49 89 2399 - 4465	Authorized Officer Ebert, W Telephone No. +49 89 2399-6016

INTERNATIONAL PRELIMINARY REPORT ON PATENTABILITY

International application No.
PCT/EP2004/050776

Box No. I Basis of the report

1. With regard to the **language**, this report is based on the international application in the language in which it was filed, unless otherwise indicated under this item.

   ☐ This report is based on translations from the original language into the following language , which is the language of a translation furnished for the purposes of:
   - ☐ international search (under Rules 12.3 and 23.1(b))
   - ☐ publication of the international application (under Rule 12.4)
   - ☐ international preliminary examination (under Rules 55.2 and/or 55.3)

2. With regard to the **elements*** of the international application, this report is based on *(replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report)*:

Description, Pages

1, 3-30	as originally filed
2, 2a	received on 08.06.2005 with letter of 06.06.2005

Claims, Numbers

1-15	as originally filed
16-20	received on 08.06.2005 with letter of 06.06.2005

Drawings, Sheets

1/8-8/8	as originally filed

☐ a sequence listing and/or any related table(s) - see Supplemental Box Relating to Sequence Listing

3. ☐ The amendments have resulted in the cancellation of:
   - ☐ the description, pages
   - ☐ the claims, Nos.
   - ☐ the drawings, sheets/figs
   - ☐ the sequence listing *(specify)*:
   - ☐ any table(s) related to sequence listing *(specify)*:

4. ☐ This report has been established as if (some of) the amendments annexed to this report and listed below had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).
   - ☐ the description, pages
   - ☐ the claims, Nos.
   - ☐ the drawings, sheets/figs
   - ☐ the sequence listing *(specify)*:
   - ☐ any table(s) related to sequence listing *(specify)*:

* *If item 4 applies, some or all of these sheets may be marked "superseded."*

Box No. V Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Yes:	Claims	1-20
	No:	Claims	
Inventive step (IS)	Yes:	Claims	
	No:	Claims	1-20
Industrial applicability (IA)	Yes:	Claims	1-20
	No:	Claims	

2. Citations and explanations (Rule 70.7):

see separate sheet

1 Documents

Reference is made to the following documents:

D1: HODES T D ET AL: "A document-based framework for Internet application control" 2ND USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS, PROCEEDINGS OF USENIX'99: 2ND SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS, BOULDER, CO, USA, 11-14 OCT. 1999, 1999, pages 59-70, XP009018011 1999, Berkeley, CA, USA, USENIX Assoc, USA

D2: ABRAMS M ET AL: "UIML: an appliance-independent XML user interface language" COMPUTER NETWORKS, ELSEVIER SCIENCE PUBLISHERS B.V., AMSTERDAM, NL, vol. 31, no. 11-16, 17 May 1999 (1999-05-17), pages 1695-1708, XP004304584 ISSN: 1389-1286

D3: US-A-5 793 368 (BEER JOHN C) 11 August 1998 (1998-08-11)

2 Objections with respect to Article 6 PCT

2.1 The term "widget configuration data", used in claim 1 (page 31, lines 9, 13 and 18) leaves the reader in doubt as to the type and purpose of this data, in particular when used in conjunction with the term "component pattern" (page 31, lines 12-13). Figures 3a and 3b seem to represent an example of the widget configuration data of the invention, and seem to suggest that the term "widget configuration data" actually relates to data for defining property settings relating to the display style of user interface components.

The following argumentation is based on the assumption that the term "widget configuration data" has the aforementioned meaning.

2 Objections with respect to Article 33(3) PCT

2.2 Document D1, which is considered to represent the closest prior art to the subject-matter of claim 1, discloses (the references in parentheses applying to this document):

A method for generating a user interface of a network node, wherein an application is structured into a core application part responsible for handling data objects and a viewer/controller application part responsible for displaying said data and initiating actions on said data (page 61, paragraph 1, left-hand column, lines 1-3),

wherein said viewer/controller application part is formed by said user interface;

a screen mask creating module for creating dynamically a screen mask of said user interface retrieves screen mask configuration data over a network which is stored on a central processing unit (page 64, section 4 *User Interface Generation*, page 66, left-hand column, lines 1-4),

in that a screen mask of said user interface is generated by said screen mask creating module, wherein said screen mask comprises at least one component (page 66, Figure 3(a)), and

in that said at least one component of said created screen mask is assigned to at least one data object and/or dynamic of said components assigned to said screen mask based upon a user action on a user interface component and/or a data object (page 63, right-hand column, paragraph 1: "*<method>-tag*").

2.3 The method of claim 1 **differs** from the disclosure of D1 in that the screen mask creating module also retrieves **widget configuration data**, in that user interface components are generated by a widget creating module using the widget data, and in that the components are stored by means of a **widget cache**.

2.4 The **problem** to be solved by this difference may be considered to be how to manipulate the visual appearance of the viewer/controller application part and how to avoid redundant operations.

2.5 However, the feature of separately providing configuration data for the visual appearance of UI components forms part of a widely-known standard specification (UIML), see for example document D2 (page 1702, Fig. 4; page 1703, left-hand column, lines 3-7: "*Various style attributes (e.g. color, font, size) for specific appliances can be specified.*"). Therefore, this feature forms part of the general knowledge and the skilled person would add it to the method of document D1 in accordance with the circumstances and without exercising inventive skill.

2.6 Likewise, the feature of using an object cache in order to avoid redundant object creation falls within the customary practice of the skilled person and has already been used in the same technical context, see for example document D3 (page 7, paragraph [0214]: "*The BEAtitude class checks ... if it has alread loaded a class into the JRE ...*").

2.7 The subject-matter of claims 14, 15, 16 and 17 corresponds in terms of product features to the subject-matter of claim 1. The objections raised in respect of claim 1 therefore apply accordingly to claims 14, 15, 16 and 17.

2.8 As to claim 2, dynamic generation of a screen mask of said user interface is disclosed by D1.

2.9 As to claim 3, changing screen mask configuration data and/or widget configuration data at least partially dynamically based upon one or more user actions is a widely used feature in user interface programs (e.g. enabling, disabling, hiding or highlighting components as a result of user actions).

2.10 As to claim 4, document D1 discloses the steps of:

retrieving said screen mask configuration (page 66, left-hand column, paragraph 1);

parsing said screen mask configuration to obtain type information about said at least one component and to obtain individual settings of said at least one component (page 64, section 4 *User Interface Generation*);

creating said at least one component by

obtaining said at least one component on the basis of at least one component pattern corresponding to said type information (page 65, left-hand column, paragraph 1; page 66, figure 3(a)); and

applying said individual settings onto said at least one component and

including said at least one component into said screen mask (page 66, figure 3(b))

2.11 The subject-matter of claim 20 corresponds in terms of apparatus-features to the subject-matter of claim 4. The objections raised in respect of claim 4 therefore apply accordingly to claim 20.

2.12 As to claims 5 and 18, document D3 discloses the additional steps of:

requesting said at least one component from a component pattern repository, which caches at least one component pattern;

identifying said at least one component pattern corresponding to said type information; and

deriving said at least one component from said at least one identified component pattern

(page 7, paragraph [0212]; page 7-8, paragraph [0214]: *"BEAtitude class"*).

2.13 As to claim 6, initialisation of the component pattern repository by:

retrieving and parsing the component configuration is disclosed in D3 (paragraph [0214]);

likewise, the steps of creating a "component pattern" and storing it in a

component pattern repository corresponds to the way of processing disclosed in D2 (paragraph [0214]: "*loading a class into the JRE*", "*calling the class' create method*").

2.14 The subject-matter of claim 19 corresponds in terms of apparatus-features to the subject-matter of claim 12. The objections raised in respect of claim 12 therefore apply accordingly to claim 19.

2.15 As to claim 7, the additional feature of statically adding at least one "component pattern" to the component pattern repository during runtime of said application is a widely used initialization technique that the skilled person would use for initializing the repository without exercising inventive skill.

2.16 As to claim 8, the feature of retrieving, identifying and parsing a component configuration is implicitly disclosed in D1, as the configuration information contained e.g. in the XML document of figure 3(a) (page 66) has to be processed in this or a similar way. Likewise, it is clear that this configuration information is used when a component is "derived" (see figure 3(b)).

2.17 As to claim 9, the additional feature of obtaining a component by cloning a component pattern merely is a selection of several known ways of obtaining an component instance which the skilled person would employ in accordance with the circumstances and without exercising inventive skill, in particular as it is provided for in the Java programming language used in the invention (e.g. interface *Clonable*).

2.18 As to claim 10, the additinal feature of adding default component configuration information to a component pattern is common practice in the field of software engineering.

2.19 The subject-matter of claim 11 seems to relate to a way of overriding default settings. This is however common practice in the field of software engineering.

2.20 As to claim 12, XML-encoded screen mask configuration data is disclosed in D1.

2.21 As to claim 13, XML-encoded widget configuration is disclosed in D3.

Annex A

17. Terminal device adapted to establish a user interface, which is operable by a user to operate an application executed by said terminal device, which comprises a screen mask creating component (240) for creating dynamically a screen mask of said user interface (GUI), comprising:

a retrieval component (260, 270) for retrieving screen mask configuration data (320) and widget configuration data (310), said data comprising ~~screen mask~~ configuration data about at least one component (10 - 18; 410),

a parsing component (250, 230, 240) for parsing said screen mask configuration data (320) to obtain type information about said at least one component (10 - 18; 410) and to obtain individual settings of said at least one component (10 - 18; 410), and for parsing said widget configuration data (310) to obtain one or more component patterns (411, 412),

a widget creating component (230 ~~240~~) for obtaining said at least one component (10 - 18; 410) on the basis of at least one component pattern (411, 412) corresponding to said type information and for applying said individual settings onto said at least one component (10 - 18; 410), and

a linking component (430) for linking said at least one component (10 - 18; 410) to at least one data object (460, 465).

16. Computer program product for establishing a user interface, wherein said computer program product comprises program code sections stored on a computer readable medium for carrying out the method of any one of the claims 1 to 13, when said computer program product is executed on a microprocessor-based device, processing device, a terminal device or a network device.

17. Terminal device adapted to establish a user interface, which is operable by a user to operate an application executed by said terminal device, which comprises a screen mask creating component (240) for creating dynamically a screen mask of said user interface (GUI), comprising:

a retrieval component (260, 270) for retrieving a screen mask configuration data (320) and widget configuration data (310), which comprises configuration data about at least one component (10 - 18; 410),

a parsing component (250, 230, 240) for parsing said screen mask configuration data (320) to obtain type information about said at least one component (10 - 18; 410) and to obtain individual settings of said at least one component (10 - 18; 410), and for parsing said widget configuration data (310) to obtain one or more component patterns (411, 412),

a widget creating component (230) for obtaining said at least one component (10 - 18; 410) on the basis of at least one component pattern (411, 412) corresponding to said type information and for applying said individual settings onto said at least one component (10 - 18; 410), and

a linking component (430) for linking said at least one component (10 - 18; 410) to at least one data object (460, 465).

18. Terminal device according to claim 17, comprising a component pattern repository (210) which caches at least one component pattern (411, 412) and from which at least one component (10 - 18; 410) is requested and an identification component (240) for identifying at least one component pattern (411, 412) corresponding to said extracted type information, wherein said widget creating component (240) is adapted to derive at least one component (10 - 18; 410) from said at least one identified component pattern (411, 412).

19. Terminal device according to claim 17, wherein said terminal device comprises as further components for initialization of said component pattern repository (210) a retrieval component (260, 270) for retrieving a component configuration (310), which comprises component configuration data about at least one component pattern (411, 412), and a parsing component (250, 230) for parsing said component configuration information, wherein said widget creating component (230) is adapted to create said at least one component pattern (411, 412) and to store said at least one created component pattern (411, 412) in said component pattern repository (210).

20. Terminal device according to claim 19, comprising a retrieval component (260, 270) for retrieving a component configuration (310), which comprises component configuration information about at least one component pattern (411, 412), an identification component (240) for identifying said component configuration information about said at least one component pattern (411, 412) corresponding to said extracted type information, and a parsing component (250, 230) for parsing said identified component configuration information, wherein said widget creating component (230) is adapted to create said at least one component pattern (411, 412) and to derive said at least one component (10 - 18, 510) from said at least one component pattern (411, 412).

design is well suited to display common parts of the same data on multiple screen masks. All controller layer application parts will act of the same model pattern such that all viewer layer application parts display the same data. In case of a field within the presentation of the data is modified the associated controller layer application part updates the model pattern such that the model layer application part notifies connected presentation elements provided by the viewer layer application part in order to effect updates presentation of the data.

It is self-explanatory that the layer design although structured into separate layers does not allow for independent implementation. Developers have to take care about consistency between the model layer application part and the connection of the viewer and controller layer application parts due to the involvement with each other. The maintaining of consistency counteracts the initial idea of a separate layered application design. That means, developers which deal with model pattern and model layer application part design, have to maintain in parallel aspects of the viewer layer application part and controller layer application part or to be more general aspects of the view and controller pattern.

A new approach to manipulating the components of a distributed Internet application with heterogeneous interfaces is described in the document "A Document-based Framework for Internet Application Control" by Todd D. Hodes and Randy H. Katz. Static XML-based documents describing required interfaces both on the client-side and on the service-side are parsed by the other party that adapts the data to the defined interface. One additional feature of solving the heterogeneous interface problem is the capability for a dynamic generation of user interfaces. Nevertheless, the disclosed framework still presents serious drawbacks in particular with respect to the description redundancy and heavy charge to be carried by the processing unit.

Another similar approach is disclosed in the document "UIML: an appliance-independent XML user interface language" by Marc Abrams et al. The XML-based language UIML (User Interface Markup Language) solves the problem of applications portability throughout a range of various appliances with very different hardware and processing capabilities (such as desktop computers, handheld devices, mobile phones, etc.). However, this solution still presents exactly the same drawbacks as the framework disclosed by the previously cited document.

The US patent 5,793,368 discloses a method and a system for displaying user interfaces and visual styles and dynamically switching between different visual styles. Users can modify or adapt the visual style of the user interfaces once displayed. A specific text-based language called UIL (User Interface Language) is used to define the user-specific elements of the user interface. The information is stored on a server and retrieved after the user's login. Once retrieved, the description is interpreted by the PGUI (Programmable Graphics User Interface) and displayed to the user. This solution suffers from the same problems, creating heavy charge to the processing unit any time a style has to be changed.

It is an object of the present invention to simplify the design and generation of user interfaces which include aspects of the viewer layer application part and controller layer application part. In particular, the user interface and to be precise a graphical user interface (GUI) is generated dynamically at runtime such that model layer application part and viewer / controller layer application part are clearly separated.

The object of the present invention is solved by a method for providing a screen mask of a user interface and a terminal device, which is adapted to perform this method.

According to an aspect of the invention for generating a user interface of a network node, whereas the user interface (GUI) is operable by a user to operate an application, the application is structured into a core application part responsible for handling data objects and a viewer/controller application part responsible for displaying said data and initiating actions on said data, wherein said viewer/controller application part is formed by said user interface, whereas a screen mask creating module for creating dynamically a screen mask of said user interface retrieves screen mask configuration data and widget configuration